Nginx; yüksek eş zamanlı çalışma kabiliyeti, yüksek performans ve düşük hafıza kullanımına odaklanılarak tasarlanmış bir Web sunucusudur.

Aynı zamanda ters vekil sunucusu, yük dengeleyici ve HTTP ön belleği olarak da kullanılabilir

SSH (Secure Shell) güvenli veri iletimi için kriptografik ağ protokolüdür.

Ssh ile ağa bağlı olan iki bilgisayar arasında veri aktarımı güvenlik kanalı üzerinden güvensiz bir ağda yapılır.

Bu durumda ağda Ssh ile haberleşen makinelerden biri ssh sunucusu diğeri ssh istemcisi olur.

**(config dosyamız)**

TCPKeepAlive yes

ServerAliveInterval 120

Host streamlit\_server

HostName 54.211.74.20

IdentityFile C:\Users\smeng\.ssh\labpem.pem

User ec2-user

---------------------------------------------------

Iden-Instructor

çalışma ortamınızda, pip install ile yüklediğimiz kütüphaneler kendileriyle beraber çalışan diğer kütüphaneleri de yüklüyor. En son dosyamızın çalıştığı bu ortamı pip freeze ile kaydediyoruz ki, ilerde versiyonlar değiştiğinde, biz kendi dosyamızı bu environment ta çalıştırabilelim

-----------------------------------------------------------------

python3 -m venv env\_streamlit

source env\_streamlit/bin/activate # Flask’da windowscular icin --> .\flask\_env\Scripts\activate

pip3 install -r requirements.txt

streamlit run app.py

*C9117-Dustin :*

* EC2 instance kur ve calistir.
* VS Code’u ac.
* VS Code sol alttan
* Open a Remote Window 'a tikla.
* Open SSH Configuration File’a tikla.
* Users/macboook/ssh/config’e tikla.
* Burada instance’in public IPv4 adresini hostname’a kopyala.
* Ayrica pem key’in path’ini IdentityFile’a kopyala ve kaydet.
* Pem anahtarini desktop’a da kopyalayip config sayfasinda onun path’ini kopyalayabilirim.
* Ardindan yine Open a Remote Window’a tikla ve cikan ekrandan Connect to Host’a tikla ve hostname adlandirma sayfasinda verdiginiz config ismine tikla.
* Baglanti saglandiktan sonra open folder ec2-user klasorune baglan.
* Yeni terminal ac. Su anda cloud’daki bilgisayara baglandin.
* Cd ile home page’e git ve burada .envs adinda bir klasor olustur.
* Ardindan cd .envs ile bu klasorun icine gir ve “python3 -m venv env\_streamlit” yaz ve enter’a bas.
* Ardindan “source env\_streamlit/bin/activate” yap.
* Ardindan bu env\_streamlit adindaki virtual environment’a kuracagimiz kutuphaneleri bir txt explorer’da bir txt dosyasi acarak icine kopyaliyoruz.
* Olusturdugumuz dosyada pakenler ve karsilarinda surumleri var, ornegin seaborn==0.11.2
* Bunlari kurmak icin venv icindeysen;
* pip3 install -r requirements.txt yapiyoruz.
* pip3 list
* ile kurulan paketleri goruntuleyebiliyorum.
* Activate yaptigim klasoru deactivate yaparak cikis yapabiliyorum.
* Bu yaptiklarimizin aynisini local’de olusturdugumuz bir klasore de yapiyoruz.

*C9639 – Muhammed :*

Hocamlar ders sonrası tekrar ederken anladığım birkaç şey var. Yazayım da hala ben gibi eksik kalan arkadaş varsa faydası olsun.

1. env olduğundan emin oluyoruz. Yoksa oluşturuyor, varsa paketleri yüklüyoruz. Ben bugün sıfırdan yapıp önce Flask sonra geçen derslerde paylaşılan requirement dosyasını (içindeki modülleri) kurdum.
2. App.py dosyasını, kod yazarken yeni işlerken kaydediyoruz mutlaka, sonra Run ediyoruz. Ben defalarca kaydetmeden run edip neden çıktısını göremiyorum dedim. Siz yaşamayın :))
3. Bazen Access hatası veriyor, birden fazla sekmede açıkmış bende ama fazlalığı kapatınca oluyordu ne hikmetse.
4. Flaskın dosyaları templates klasörü içinde oluyormuş. Index ve result dosyalarımız vardı hatırlarsınız. Bir de static dosyalar vardı, onlar da static klasöründe olacakmış. Flaskın defaultu böyleymiş
5. environment kurmak ve klasörler arası navigasyon için birkaç tanecik terminal kodu var toplamda, onları not defterine yazın githuba atın sonra kullanmak için bakarsınız.

23.02.2022

sudo yum install tmux  
   16  tmux new -s flask\_server  
   17  exit  
   18  python3 app.py  
   19  cd usedcar\_project/  
   20  tmux ls  
   21  tmux kill-session -t flask\_server  
   22  tmux new -s flask\_server  
   23  exit  
   24  cd usedcar\_project/  
   25  tmux ls  
   26  tmux attach -t flask\_server

24.02.2022 DOCKER ( Class-chat-en)

2. sudo yum update -y

# Install the most recent Docker Community Edition package.

3. sudo amazon-linux-extras install docker -y

# Starting a docker service

4. sudo systemctl start docker

# Enable docker service so that docker service can restart automatically after reboots.

5. sudo systemctl enable docker

# Check if the docker service is up and running.

6. sudo systemctl status docker

# Check the running containers

7. sudo docker ps

# - Add the `docker` group to the `ec2-user` to run docker

# commands without using `sudo`.

8. sudo usermod -aG docker ec2-user

# - Normally, the user needs to re-login into bash shell for the group `docker` to be effective, but `newgrp` command can be used activate `docker`

# group for `ec2-user`, not to re-login into bash shell.

9. newgrp docker

# from now on we don't have to type sudo

# check running containers again

10. 'docker ps' --> same command with 'docker container ls'

# show runnig and stopped containers

11. docker ps -a

# docker image list

13.docker image ls

# pull an image --> show from docker hub as well --> tags etc

14. docker pull ubuntu # --> latest is the default tag

16. docker info

# Create a container from an image

# only required argument is image name --> ubuntu

# if the image does not exist on the machine, it is pulled from the registry

17. docker run -it --name myubuntu ubuntu # --> docker container

# -it --> opens a virtulal terminal session in docker container

# - interactive terminal

# container icinde python olmadigini goster

18. python # in container

19. apt-get update -y

20. apt-get install python3

21. --> check python versions in docker container and in linux sehll in another terminal

22. --> try to import numpy in python in docker

21. apt-get install python3-pip

22. python3 and

23. --> import numpy as np

24. terminali kapat --> container will be gone

25. docker ps

26. docker ps -a --> will show the exited container

# REMOVING A CONTAINER

26. docker rm XXX # XXX --> CONATINER ID's first 3 chars

docker rm container name --> same thing

27. docker ps -a --> no container w same name

28. history --> summarize all the commands we had so far

# Detach mode

# Create a sh to run commands -->docker\_runner.sh

#!/bin/sh

sudo yum update -y

sudo amazon-linux-extras install docker -y

sudo service docker start

sudo usermod -aG docker ec2-user

newgrp docker

systemctl enable docker

27. sh docker\_runner.sh

28. --> starta a new terminal and sh docker\_runner.sh

28. docker run -it -d --name myubuntu ubuntu --> this command does not attach the container so we cannot go inside the conatiner immediately but docker run -it --name myubuntu ubuntu --> attach the doctainer immediately

29. docker ps

# Going inside the docker conatiner

1. docker exec -it xxx bash

2. docker attach xxx

# Exit edilmis bir conatineri caslistirma

docker start xxx

dcoker attach xxx

# Exit Container

35. 'CTRL + D' or 'exit'

36. docker ps

37. 'docker start xxx' and ('docker exec -it xxx bash' or 'docker attach xxx')

40. docker stop xxx

39. docker start xxx

38. history --> summarize what we have done so far

23. apt-get install vim -y

39. ls -l --> list files in docker container

41. docker image ls

# Remove Docker image

41. docker rmi xxx --> cannot remove stop containers's images

# --rm flag --> removes a container when it stops

41. docker rm xxx

42. docker run -it -d --rm --name myubuntu ubuntu

43. docker ps

44. docker attach xxx

45. --> exit the container #--> exit

46. docker ps -a # --> container is gone

# VOLUME MOUNTING

47. docker run -it -d --rm --name myubuntu ubuntu

48. docker attach xxx

49. apt-get update -y

50. ls

51. mkdir my\_data && cd my\_data

52. apt-get install vim -y

53. vim my\_test\_file --> hello dear friends. This is inside docker --> ESC :wq

54. cat my\_test\_file

55. ls

56. exit # --> exit the container

57. docker ps -a # --> nothing is presistent any more

58. docker run -it -d --name myubuntu ubuntu

48. docker attach xxx

49. apt-get update -y

50. ls

51. mkdir my\_data && cd my\_data

52. apt-get install vim -y

53. vim my\_test\_file --> hello dear friends. This is inside docker --> ESC :wq

54. cat my\_test\_file

55. ls

56. exit # --> exit the container

57. docker ps -a --> we see the container

58. docker start xxx

59 ls --> and show the file, still exists. but it is good as long as the contianer survives

59. docker container inspect xxx

58. docker volume ls # deleting volume is docker volume rm my\_volume\_name

60. docker volume create MY\_HOST\_DATA # --> creates in host (EC2)

61. docker volume inspect MY\_HOST\_DATA # copy the Mountpint

62. sudo ls /var/lib/docker/volumes/MY\_HOST\_DATA/\_data

62. docker run -itd -v MY\_HOST\_DATA:/data\_in\_docker ubuntu

63. docker attach xxx xxx

64. apt-get update -y

53. apt-get install vim -y

54. cd data\_in\_docker

53. vim my\_test.text --> hello from Docker class

54. cat my\_test

55. exit docker

56. docker attach xxx

57. cd data\_in\_docker # --> data still exists

58. exit

59. docker stop xxx & docker rm xxx

58. docker volume ls

60. docker volume inspect xxx

57. sudo su

60. sudo ls /var/xxxxxx # volume location

60. sudo cat /var/xxx/my\_test.txt

Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance. When you launch an instance, you can specify one or more security groups. If you don't specify a security group, Amazon EC2 uses the default security group.

AWS’de portları açmak için;

1. Instances kısmından instance’ımızın Security kısmına gidiyoruz.
2. Ordan security groups’a,
3. Ordan edit inbound rules’a gidiyoruz.
4. Sol alttan Add rule diyoruz.
5. Port range 5000, 80 ve 81 giriyoruz. Source olarak da Anywhere IPv4 diyoruz.
6. Sonra Save rules diyoruz.

Ve artık hazır.

**Not** : Eğer add rule’da type olarak “all traffic” dersek başka bir rule eklememize gerek kalmaz fakat bu çok sağlıklı bir şey değil.

**Kendi Projemde çalıştırdığım kodlar**

1. python3 -m venv tenv
2. source tenv/bin/activate
3. pip install pandas streamlit scikit-learn xgboost
4. pip3 install xgboost --no-cache-dir
5. pip install --upgrade pip
6. pip install pandas streamlit scikit-learn
7. pip freeze
8. sudo yum -y install tmux # tmux yaptığımız zaman vscode kapatsak bile link çalışmaya devam eder.
9. tmux new -s mysession # ama her düzenlemek istediğimizde tekrardan tmux yapmamız lazım.

\*9’da ctrl b ve ctrl d yaptık ordan çıkmak için. (D’ye basarken elimizi çekiyormuşuz)

:slightly_smiling_face:Dersteki projede çalıştırdıklarımız(ın bazıları) - 26.02.2022

* conda deactivate
* python3 -V
* python -m venv senv
* python3 -m venv senv
* source senv/bin/activate # windowsta 🡪 . senv/Scripts/activate
* pip install -r newrequirements.txt
* streamlit run streamlit\_student\_app.py



